STRUCTURAL DOMAIN MODELING FOR UNDERSTANDING EQUIPMENT FAILURE MESSAGES

Kenneth Wauchope

Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory, Code 5512
Washington, DC 20375-5000

Abstract

The goal of natural language understanding computer systems is to analyze and make use of the information contained in English or other natural language discourse. Understanding texts that discuss complex pieces of equipment, such as Navy equipment failure reports (CASREPs), requires the possession not only of general knowledge about the types of objects and predicates in the domain, but also detailed knowledge about the particular equipment in question. This more expert level of knowledge is needed to dereference the names and descriptions of equipment referred to in the text, and to infer their causal relations and operational states when these are only implicitly expressed by the message writer. Knowing the structural configuration of the equipment is useful in both tasks, and a structural domain model can be extracted readily from equipment manuals and their accompanying parts lists, thus easing the "knowledge acquisition bottleneck" problem and making practical applications more feasible.

Introduction

CASREPS (Casualty Reports) are Navy messages reporting shipboard equipment failures and attempts at their diagnosis and repair. Although consisting primarily of information formatted into fixed fields, these reports also include narrative English text describing and amplifying on the formatted portion of the message. The informational content of these narrative portions is typically unavailable to conventional data processing systems.

The goal of natural language understanding (NLU) computer systems is to analyze and make use of the information contained in English or other natural language discourse. At the Navy Center for Applied Research in Artificial Intelligence, we have been addressing the task of analyzing the free text in CASREP messages and formatting it into an application-neutral form suitable for inferencing, information retrieval, and other Artificial Intelligence (AI) tasks. We take a computational linguistic approach to this analysis, subjecting each input sentence in turn to lexical lookup, parsing against a broad-coverage English grammar, enforcement of domain semantic constraints, syntactic regularization, and finally mapping to a framelike (i.e. slot-filler) semantic representation.

The application task we have been exploring in recent years is the automated highlighting of important information in the narrative portion of CASREPs. The Navy employs a team of contractors whose task is to generate a short textual extract of each CASREP message for use in failure trend analysis, and our goal was to attempt to extract the same information by automated means. Since the human-generated extracts rarely contained text that was not present in the original narrative, but usually restated one or more clauses selected from the message, we took the same approach. Our system modeled "importance" as a series of saliency ratings on such information types as causality, results of investigation, and malfunction. Those clauses containing the greatest amount of salient information were chosen for inclusion in the extract.

Our first prototype system, called SUMMARY [6,7], could extract only *sentence-local* causal and failure information — that is, information recoverable from lexical and syntactic content. Hence the system could recognize the lexical item *erratic* as connoting a failure, and could normalize the syntactic structure *X be due to Y* into *Y cause X*. It also contained a few rules for extracting such information by inference (again only on a local basis), such as recognizing that *X impaired Y* entails that Y is in a failed state and that X caused that failure to occur, and then inferring that X must be in an undesirable state as well.

This work was supported by the Office of Naval Technology under Artificial Intelligence Project RS34C74 and by the Office of Naval Research under Natural Language Research project RR015-08-01.

SUMMARY could not make use of intersentential context in performing its inferences, however. One important role of narrative discourse structure is to implicitly convey causal relationships between assertions. The message writer seeks to communicate a coherent account of a series of related events, and the reader uses knowledge of the domain to recover the causal relations implicit in the sequence of statements contained in the narrative. Such domain knowledge is often represented in NLU systems as *scripts* [9], which are stereotypical descriptions of common event sequences. To establish these interconnections the reader must also use general principles of discourse structure, to resolve the referents of pronouns, decide when two noun phrases are referring to the same entity, and analyze sentence fragments and other elliptical phenomena.

The only causal relationship that SUMMARY deemed important to include in the CASREP extract was the causing of one equipment malfunction by another. Such events tend to be highly idiosyncratic rather than stereotypical, so scripts are not an appropriate knowledge representation for their analysis. We also do not believe that a generic description of various types of equipment and their functionality suffices for the kind of inferencing needed. Instead, we feel that the reader of an equipment failure message must rely on a certain amount of expert knowledge of the structure, operation and behavior of the particular equipment in question. Our next version of the text extraction system, called TERSE [10], included a model of the piece of equipment (a diesel-driven air compressor used to start a gas turbine engine) that was the topic of our CASREP message set. TERSE also contains routines for using the model to analyze complex part names and descriptions, resolve multiple references to the same piece of equipment, and confirm implicit discursive causal relations. The system was built in Intellicorp's KEE system development shell and runs on a Symbolics LISP machine and a Sun workstation.

Much has been said about the "knowledge acquisition bottleneck" that is the bane of large knowledge-based systems. The Starting Air Compressor (SAC) discussed in our message set is only one of hundreds of pieces of shipboard equipment for which CASREPs are issued. If TERSE was to be a system the Navy could truly use, then the domain models for all this equipment would have to be kept reasonably simple. In this light, we wanted to experiment with whether a strictly structural or topological equipment model — which could be constructed directly from equipment manuals and accompanying parts lists — would be useful in our message extraction task. This paper describes the structural equipment model we developed for TERSE, the interaction of domain and discourse knowledge in the dereferencing of equipment part names and recovery of causal/state information, and an informal evaluation of this approach's merits and drawbacks.

Contextual Knowledge

Message 1 in Fig. 1 gives the complete text of a CASREP narrative remark of medium length. Our pilot system SUMMARY (without an equipment model) chose the last sentence *Splines were extensively worn* as its extract, since that sentence contains a part name (*spline*) in association with a lexical item (*worn*) that clearly connotes damage to the part. The contractor-generated extract contains that sentence as well, but also includes the preceding sentence *Drive shaft was found to rotate freely at the SSDG end.* SUMMARY did not recognize *rotate freely* as connoting a failure, since on a sentence-local basis the semantics of *freely* are ambiguous — indeed, human subjects we presented with that sentence out of context disagreed whether it represented a positive or negative condition. It is only in the context of the final sentence of the message that *rotate freely* can be disambiguated: the shaft's splines are worn, erosion of a moving part can result in malfunction, and therefore rotating freely is (in this particular context) a failure.

Several types of contextual knowledge are required to understand the implications of this message. First there is domain-specific **world**

Message 1
SAC received high usage during two BECCE periods. *CCS received a report that LO pressure was dropping.* *Alarm sounded.* *Loud noises were coming from the drive end during coast down.* *Drive shaft was found to rotate freely at the SSDG end.* *Splines were extensively worn.*
Message 2
Starting air regulating valve failed. *Unable to consistently start nr 1b gas turbine.* *Valve parts excessively corroded.*
Message 3
While conducting operational checks, experienced engagement malfunction. *Discovered that the engine drive adapter hub internal splines had sheared, causing the SAC to fail to engage.* *Spline shaft rcvd w/assist from DESRON 8.* *Installed on #3 SSDG.* *Tested sat on 25 Feb.*
Message 4
Loss of lube oil pressure during operation number 2 SSDG. *Metal particles found in lube oil filter.*

Fig. 1. Example SAC CASREP narratives

knowledge, needed to identify the components of the SAC that can be described by the noun *shaft* and determine which of these can be referred to by the more specific compound nominal *drive shaft*. It also includes the knowledge that splines are a physical feature of many different pieces of equipment like shafts and gears, that there is more than one alarm in the system, that the trigger condition for one of those (the lube oil alarm) is a drop in oil pressure, that one drive shaft has splines and is located at the SSDG end of the SAC, and that lube oil pressure is maintained by an oil pump that receives its power indirectly from that shaft.

Second, there is the discourse knowledge that *the drive end* and *the SSDG end* are elliptical references to physical locations on some object described earlier in the message. Discourse principles also predict that a splined object must have been referred to earlier in the message to resolve the referentially ambiguous *splines* of the last sentence, and that the message context can similarly be used to disambiguate the noun phrases *alarm* and *drive shaft*.

Finally there is the scriptal knowledge that the narrative message is describing a sequence of related events, usually time-ordered: background history (sentence 1 of the example), failure events (sentences 2-3), and findings (sentences 4-6). Descriptions of event history tend to preserve order of causality: the drop in lubricating oil pressure caused the alarm to sound. Troubleshooting descriptions, however, usually proceed from intermediate findings to specific causes: there were noises, which turned out to be caused by the shaft rotating freely, which in turn was found to be caused by worn splines.

The equipment model is the world-knowledge source for TERSE, and routines for noun phrase dereferencing and semantic disambiguation constitute the discourse component. Finally, some of the heuristics we use in inferring intersentential causal implications can be considered an extremely simple form of scriptal knowledge, but aside from that, the system does not yet do script analysis or temporal processing. We will explore each of these in turn.

The Equipment Model

The TERSE equipment model is constructed using *frames*, or typed slot-filler structures organized into an inheritance hierarchy. Each slot represents either an attribute of the piece of equipment, or a relationship of the equipment to other components. Many frame-oriented natural language understanding systems use a technique of dynamic instantiation to interpret the meaning of phrases. For example the noun phrase *lube oil pump shaft* might trigger the creation of new instances of the **shaft**, **pump** and **lube-oil** frames, with a pointer from the *:supercomponent* slot of the first instance to the second, and a pointer from the *:processed-substance* slot of the second to the third. The resulting representation is equivalent to the generic knowledge about equipment that we ourselves had when first beginning to read and try to understand SAC CASREPS: oil can lubricate, shafts can be part of pumps, and pumps can process oil. Without specific knowledge about the way in which the various components of the compressor are interconnected, however, we found it difficult to understand many of the causal implications of the messages.

A second approach — the one we have taken in the present work — is to build *a priori* a completely instantiated model consisting of the components known to comprise the equipment in question, derived from equipment manuals and parts lists. The process of understanding the noun phrase *lube oil pump shaft* then consists of exploring the model for a configuration of frame instances that has the slot-filler structure just described. Such a model can incorporate specific information about how this particular pump fits into the overall architecture of the compressor and other associated equipment. This level of knowledge corresponds to the expertise of a reader familiar with the particular domain, and permits increased and more accurate inferencing.

The drawback to our approach is that a new equipment model must be constructed in detail for each new domain of CASREP being processed. Once frames have been developed describing the various generic components (such as pumps and shafts), however, a structure editor or similar front-end tool can make the instantiation process less painful. In constructing the Starting Air Compressor model, we found a need to incorporate only 85 of the 350 individual components listed in the Navy equipment manuals. Those 85 included all the primary functional elements of the compressor, bottoming out at indivisible parts such as gears, hubs and shafts. The remaining 265 items were low-level accessory items like pins, bolts and washers that did not contribute substantially to a basic functional understanding of the equipment's behavior. Fig. 2 illustrates two nodes in the SAC equipment model, linked to each other via their *:input-from* and *:output-to* slots.

Unit **<sac>** =
:name *shaft driven air centrifugal compressor*
:isa **compressor**
:partno *681950-1-1*
:aka *SAC*
:input-from **<ssdg>**
:output-to **<gas-turbine-engine>**
:processed-substance **<starting-air>**
:supercomponent **<starting-air-system>**
:components **<start-air-valve>, <oil-supply-section>...**

Unit **<gas-turbine-engine>** =
:name *gas turbine engine*
:isa **turbine-engine**
:partno *LM2500*
:aka *GTE, GTRB, GTM*
:function *propulsion*
:input-from **<sac>**
:supercomponent **<propulsion-system>**
:fuel **<natural-gas>**

Fig. 2. Sample Frame Instances

Noun Phrase Dereferencing

The first task in using an equipment model for natural language understanding is to establish a link from each equipment-naming phrase in the message to the equipment model node it denotes. In a technical domain like mechanical equipment, complex nominals frequently occur, composed from various combinations of premodifiers (typically adjectives and nouns, but sometimes possessives and numbers) and postmodifiers (usually prepositional phrases), for example

ship's turbine start air system
muffler assembly body end flange
discharge hose from surge air valve assembly
low speed coupling from diesel to SAC lube oil pump
number 4 SAC 21 inch woven cotton covered rubber surge air hose

It is also clear from the message data that writers often do not refer to pieces of equipment by their proper Navy-manual names, but instead use a variety of descriptive naming conventions, as seen in the following references to the **drive shaft** part:

shaft *drive shaft*

SAC shaft	*compressor shaft*
input shaft	*spline shaft*
SAC drive shaft	*SAC input shaft*
input spline shaft	*input drive shaft*
spline drive shaft	*SAC input drive shaft*
splined input drive shaft	*SAC spline input drive shaft*
connecting shaft	*581732-1*

It would not have been wise to attempt to provide TERSE with synonym lists for each part, since new data would be likely to present unanticipated new references, and system robustness was a top priority.

In general the problem of noun phrase analysis is a difficult issue. This is largely because of the prevalence in such phrases of "vague predicates" such as the N-N or noun-noun relation (*muffler assembly body end flange*) and ambiguous prepositions like *with* and *from* (*discharge hose from surge air valve assembly*). These relations cannot be successfully analyzed in general predicate-argument terms, but require world knowledge to be understood.

In a constrained domain, however, the problem is greatly simplified since the number of choices is much reduced. Left modifiers in equipment names fill only a limited set of semantic roles, and it is these roles that the slots of the model units capture. For instance, since there are at most six ways in the TERSE equipment model in which two units can be linked together, there are correspondingly only six ways in which one part name can be an N-N modifier of another: as its *:supercomponent* (pump shaft), *:components* (spline shaft), *:input-from* (diesel hub), *:output-to* (ring gear hub), *:processed-substance* (oil pump), or *:fuel* (gas turbine). The *a priori* modeling approach simplifies matters even more since it requires that a noun phrase be more than just semantically well-formed (capable of instantiation) to be accepted, but match an existing configuration in the model. Although it is reasonable that the phrase *ring gear hub* could refer to a gear hub that is ring-shaped, for example, no such piece of equipment exists in the SAC model, permitting that analysis to be rejected.

Noun phrase dereferencing proceeds in two stages in TERSE. First, the noun phrase is structurally analyzed by a parser that, interleaved with the equipment model, incrementally develops a set of candidate referents for the components of the phrase. Second, if the fully analyzed noun phrase turns out to be ambiguous (has a referent set containing more than one candidate), disambiguation techniques are used to select the most acceptable of the choices.

Structural Analysis

A semantic grammar (one defined using domain-specific categories rather than general lexical and syntactic ones) is used in the structural analysis of equipment nominals, to take advantage of natural constraints in the domain. Each word in the CASREP lexicon is assigned to one or more semantic word classes that have been derived semi-automatically by distributional analysis of a corpus of message texts [8]. For example, in the SAC CASREP domain the verb *reveal* tends to co-occur with words like *investigation*, *inspection*, *troubleshooting*, and *test* in Subject position (as in the sentence *Investigation revealed no SAC rotation*), so this set of words is assigned the semantic category **invest**, all referring to investigatory acts of some sort. The semantic word classes **part**, **func**(tion), **prop**(erty), and **status** serve as the preterminals of the equipment-name grammar, shown in Fig. 3. For example this grammar will analyze the phrase *lube oil pump* as a UNIT in two different ways:

```
(EQUIP
  (ATOM (*FUNC lube))
  (EQUIP (UNIT (EQUIP (*PART oil)))(EQUIP (*PART pump)))))

(EQUIP
  (UNIT (EQUIP (ATOM (*FUNC lube))(EQUIP (*PART oil))))
  (EQUIP (*PART pump))))
```

where in the first analysis *lube* incorrectly modifies *oil pump*, whereas in the second analysis it correctly modifies *oil*.

The structural analysis is done by an augmented context-free syntactic analyzer, the PROTEUS parser [3]. The augmentations take the form of *restrictions* that test for various well-formedness conditions whenever particular nodes are constructed. In TERSE, the restrictions enforce semantic constraints by consulting the equipment model during the parse to see if the node just composed can be interpreted as one or more units in the model. If successful, the node is assigned a node attribute called *referent-set* containing pointers to those units, and the node is added to the parser's working memory for further possible use; if unsuccessful, the node is pruned from the parser's search space and never needs to be reconsidered.

```
<UNIT> ::= <EQUIP> | <PROPERTY>.
<EQUIP> ::= <*PART> | ( <ATOM> | <UNIT> ) <EQUIP>.
<PROPERTY> ::= <*PROP> |
          ( <EQUIP> | <*STATUS> ) <PROPERTY>.
<ATOM> ::= <*FUNC> | <*STATUS>.
```

Fig. 3. Equipment Nominal Grammar

Three of TERSE's most useful restrictions (and the grammar productions after which they are applied) are as follows:

1. VALID-EQUIP (after *EQUIP ::= *PART*):
   loop for unit in model do
     if either unit is in class PART
     or the part-number of unit is PART
     or an acronym of unit is PART
     then add unit to the *referent-set* attribute of the present node
   if *referent-set* return true else return false

2. VALID-ATOM-MODIFIER (after *EQUIP ::= ATOM EQUIP*):
   loop for unit in *referent-set* of element EQUIP do
     if some slot of unit contains ATOM
     then add unit to the *referent-set* attribute of the present node
   if *referent-set* return true else return false

3. VALID-UNIT-MODIFIER (after *EQUIP ::= UNIT EQUIP*):
   loop for unit-1 in *referent-set* of element UNIT do
     loop for unit-2 in *referent-set* of element EQUIP do
     if either unit-2 is a subcomponent of unit-1
       or unit-2 is-directly-connected-to unit-1
       or some other slot of unit-2 contains unit-1
     then add unit-2 to the *referent-set* attribute of the present node
   if *referent-set* return true else return false

Restriction 1 represents the simplest case: it accepts *LO* and *23699* as minimal equipment names for the **<lube-oil>** unit because they are respectively *:aka* (acronym) and *:partno* values of the unit, and it interprets *oil* as the referent set {**<lube-oil>**, **<diesel-oil>**} because both those units have **oil** as an *:isa* ancestor in the sort hierarchy. Restriction 2 analyzes *lube oil* as unit **<lube-oil>** by taking the referent set of the head noun *oil* (found by Restriction 1 earlier in the parse) and keeping only that member of the set that has *lube* as a *:function* slot filler, thus discarding **<diesel-oil>**. Restriction 3 accepts *pump shaft* because one model unit matched earlier by the head noun *shaft* has a *:supercomponent* link to one unit (**<oil-pump>**) matched by the modifier *pump*, and that shaft (**<oil-pump-shaft>**) becomes the referent of the compound nominal.

Recursive application of these rules allows a wide variety of nominals to be understood as references to the same piece of equipment. For example the frame instances shown in Fig. 2 understand all the following ways of referring to the starting air compressor:

compressor	*681950-1-1*
SAC	*air compressor*
starting air compressor	*GTRB SAC*
gas turbine SAC	*LM2500 start air compressor*

Because the grammar allows the restrictions to incrementally reject subphrases of a noun phrase during the parse, the search space is pruned in an efficient manner. While parsing the noun phrase *lube oil pump shaft*, for example, the rejection of the incorrect ("lube oil-pump") analysis of the first three words of the phrase means that node is no longer available to conjoin with the head noun *shaft*, so one less parse of the entire phrase needs to be generated and tested. Based on the unaugmented grammar alone, there are 14 possible parses of the complete phrase *lube oil pump shaft*, such as "a shaft that pumps lube oil" (just as *start air check valve* is a valve that checks start air). By the time the augmented parser is ready to build analyses for the entire phrase, however, twelve of the 14 have already been ruled out by earlier consultations of the model.

Contextual Disambiguation

Frequently the model-based interpretation process described in the previous section returns an ambiguity set of several possible referents for a noun phrase. This is the result of underspecification by the message writer, who expects the reader to use other information in the message context to resolve the reference. This information generally consists of prior references, either to the same or to semantically related entities. In Message 2 of Fig. 1, the valve in the underspecific phrase *valve parts* is intended to be coreferential with the entity that was more fully specified in the first sentence of the narrative, *starting air regulating valve*. Similarly, in Message 1 of Fig. 1 the underspecified name *alarm* is expected to be understood as the *lube oil pressure alarm* and the name *splines* as *drive shaft splines*, in both cases based on semantic information in the sentence preceding the reference.

Coreference is the simpler phenomenon to handle: roughly speaking, if one member of an ambiguous referent set is found to have already been referred to earlier in the discourse, then it can be chosen as the object of the ambiguous reference. Resolving the second type of underspecification requires that the prior referent be only semantically related to (not identical with) the ambiguous one. The maintenance of a *focus list* [1] is a computational linguistic method often used for dealing with such referential ambiguities. This is a list of entities that have already been referred to in such a way that they can serve to resolve later ambiguous references. Techniques such as spreading activation or marker-passing [4] can be used with the focus list to determine semantic relationships between the prior and later referents.

We initially chose to design a more heavily model-based rather than linguistically justified disambiguation algorithm for TERSE. In our approach, ambiguous references can be resolved not only by unambiguous prior referents (either identical or related), but with the aid of ambiguous ones, and indeed by *later* referents as well, both ambiguous and unambiguous. This is done by generalizing the process to one of nearest-neighbor clustering in the equipment model network. Just as the links in the equipment model are used to interpret individual noun phrases, they can also be used to establish semantic relations between candidate referents. The algorithm is as follows.

Each candidate in each ambiguity set is compared pairwise with every other unit referred to in other clauses of the message, even those that are themselves ambiguous candidates. For each of these pairs, the model is traversed to determine if the two units are connected by any combination of links (such as *:input-from* or *:components*). If such a chain of one or more links is found, a certainty metric associated with each candidate is incremented by the inverse of one plus the length of the chain (identity counts as a chain of length zero), so that candidates closely connected in the model to other referents have their scores boosted more than those distantly related. In addition, the algorithm weights the clauses by how widely separated they are from each other in the message text. For instance, a semantically related object referred to in the clause immediately before a candidate will be given greater consideration than one mentioned earlier. At completion, the candidate with the highest certainty score in each ambiguity set is chosen as the referent of its noun phrase.

A simple example is illustrated in Message 3 of Fig. 1. In this instance all part references (*engine drive adapter hub internal splines*, *SAC*, and *SSDG*) are unambiguous except for *spline shaft*, which matches four objects in the model. The disambiguation routine generates the score matrix shown in Fig. 4, where each row is a candidate referent for *spline shaft* and each column is a unit referred to in the message. The **<drive-shaft>** candidate gets the highest score, primarily because it is more closely linked in the equipment model to the <SSDG> and **<drive-adapter-hub-spline>** units than are the other three shafts. Had one or more of the other referring expressions in the message also been ambiguous, a row and column for each additional candidate referent would have been provided as well.

	SPLINES	SAC	SSDG	*TOTAL*
DRIVE SHAFT	0.83	0.83	0.83	2.50
GEAR SHAFT	0.50	0.83	0.64	1.98
COUPLING DRIVE SHAFT	0.47	0.83	0.62	1.93
WHEEL SHAFT	0.40	0.75	0.57	1.72

Fig. 4. Disambiguation score matrix

Model-Based Inferencing

Applied to Message 2 of Fig. 1, the dereferencing procedures have disambiguated *starting air regulating valve* in the first sentence by finding that one of those valves has a short chain of *:output-to* links to the **<gas-turbine>** referenced in the second sentence, and they have also determined that *valve parts* in the final sentence refers to *:components* of the aforementioned valve. The part-whole and functional-connectivity links that were used in both equipment nominal interpretation and disambiguation are now used in recovering implicit causal and state information from the message.

Since the valve, its components, and the engine are all known to be in a negative state, the reader is led to suspect that the writer is implying causal relationships among the three predications. Discourse structure alone cannot determine the flow of causality from one predication to the next, since sentence order in CASREPs can connote either order of failure events (forward flow) or order of investigation findings (typically backward flow). The reader instead must rely on domain knowledge to make the inferences, and TERSE attempts to duplicate this behavior through four heuristics.

The first heuristic involves the partitioning of state-predicating words into an ordered sequence of three groups, which we call **damage**, **failure** and **loss**. **Damage** includes such words as *corrode, shear, wear,* and *erode*. **Failure** words connote decreased equipment functionality and include *low, slip, drop* and *seize*. Finally, **loss** connotes loss of equipment usage by ship personnel and includes words like *loss, inoperative* and *unable*. Our test data suggest that inferable causality in CASREPs flows from members of one group to members of the same or a subsequent group, but not the other direction. Hence it is more likely that the corrosion of the valve parts (**damage**) caused the failure of the valve (**failure**) than the other way around, and that each of these in turn caused the inability to start the engine (**loss**). One can certainly devise counterexamples to this heuristic, but in our CASREP message corpus the rule works remarkably well nonetheless.

The second heuristic prefers flow of malfunction causality from component to supercomponent (assembly) rather than the inverse. This again leads to the inference that the corrosion of the valve parts caused the valve failure. Again one can think of counterexamples such as a failed valve allowing salt water in to corrode its parts, but since the flow of causality from assembly to component is so much rarer than its opposite, in such cases the writer would probably not permit the causality to remain merely implicit in discourse structure but would spell it out explicitly. In our SAC CASREP corpus of 182 messages there is only one instance in which the malfunction of a part (the slippage of a drive shaft) resulted in damage to its components (grinding down of its splines), and that causal relationship was spelled out explicitly — otherwise the simpler inference would have been that the ground splines caused the slippage.

The third heuristic prefers flow of malfunction causality from functional input to functional output. This leads to the inference that the valve failure caused the inability of the turbine to start, since the latter piece of equipment can be reached from the former by a series of *:output-to* links. The counterexample to this heuristic in the equipment world is a **blockage** state in which flow of energy or material is impeded by a failure downstream, causing it to "back up" and cause failure or damage upstream. Since such situations do occur with some frequency in the SAC CASREP domain, blockage words like *seize* and *clog* have to be given special consideration in the system.

The fourth and final model-based inferencing rule is used in the disambiguation of equipment status. The only CASREP datum we have found so far that requires this rule is the word *freely* in the Message 1 of Fig. 1, so the rule is still *ad hoc* and probably requires refinement. At present the rule states that if a clause containing an ambiguous status word can be inferred to be both a cause and an effect of other malfunctions stated in the message, then assume that the clause does connote a malfunction itself. Since interpreting *drive shaft rotate freely* as a malfunction allows it to be seen both as a result of the worn splines and a cause of the oil pressure dropping (since there is an *:output-to* chain from the shaft to the oil pump), the rule decides that *freely* should be interpreted as a malfunction.

It must be kept in mind that the function of these heuristic rules is to recover causal and state information deliberately implied by the message writer, not to do fault diagnosis (i.e. hypothesize causal relations that the message writer does not believe and is not trying to convey). The writer's

motivation for leaving certain information implicit is that the inferences the reader must use to recover that information are *easy* ones, at least if the reader is sufficiently expert. This is one of our main reasons (the other is the knowledge acquisition problem) for experimenting with simple modeling techniques and inferencing rules.

Discussion

The TERSE equipment model is strictly structural or "topological" in that it does not attempt to model equipment *behavior* per se, either qualitatively or quantitatively. The model knows that the Starting Air Regulating Valve processes air, that its function can be referred to as *regulate*, and that the valve's output goes to the gas turbine engine. However it does not know what the token "regulate" means, what the valve's behavior is (to partially open or shut), or what attributes the valve imparts to the air (increase or decrease in volume/pressure). The model only assumes that if the valve malfunctions in any way, the valve's output will also "fail" somehow, causing both its parent supercomponents as well as the next component downstream to have impaired functionality.

Although this approach was satisfactory because of constraints peculiar to the SAC CASREP domain and the application task, we increasingly felt the need to add pieces of behavioral knowledge to the system to solve problems such as blockage events (flow of malfunction from output to input) and message references to specific attributes (such as temperature and pressure) that the equipment imparts to its processed substances. Had the SAC contained feedback loops, behavioral knowledge would also have become necessary to decide which component of the loop was being claimed by the message writer to be the initial source of a malfunction: the overheating of an engine might conceivably cause a fan belt to slip, but it is much more plausible that a loose fan belt would cause an engine to overheat. In short, knowing the *kind* of malfunction that has occurred is often critical in determining its possible consequences.

Recent related work [5] has taken the approach of developing a full *simulation* model for text understanding. Such a model can be used not only in the information extraction task we have been addressing but also other aspects of language understanding, as well as interfacing with diagnostic or tutorial systems. Our approach instead concentrates particularly on trying to represent the knowledge that an expert reader possesses about equipment, rather than a model of the equipment itself. Of course, since machinery is a human artifact, a simulation of a machine's behavior can to a great extent mirror the knowledge that an expert has about that behavior. The need to add behavioral information to our model was certainly leading us in the direction of qualitative modeling [2], although a full simulation model was more powerful than was needed for our application. For example, given Message 4 of Fig. 1, a simulation model would still not be able to resolve whether the loss of pressure was caused by the particles clogging the filter, or by mechanical damage elsewhere in the system (which the particles are evidence of), since the writer does not seem to be making a clear claim for either interpretation.

The disambiguation algorithm we developed may be more generalized than is necessary, and on large messages the pairwise comparison of referents might become prohibitively expensive (this has not proven to be the case so far, however). The use of a focus list, and constraining the context to backward references (and a limited class of forward references), may suffice to resolve ambiguous equipment nominals in CASREPs. For example, traditional discourse principles would immediately accept *starting air regulating valve* in Message 2 of Fig. 1 as an unambiguous coreferent for *valve* later in the message, whereas our algorithm gives the possible coreference a high score but then continues to accept other disambiguating contextual information as well.

Summary

We have tried to show that domain-specific knowledge is required to recover implicit references to causality from narrative texts that describe equipment failures. Generic knowledge about equipment is not enough, since expert knowledge of the architecture of the particular machinery under discussion is needed to trace the flow of functionality (and thus malfunctionality) from one component to another. The same body of domain knowledge can be used in the dereferencing of equipment-denoting noun phrases (both local semantic interpretation and disambiguation based on discourse context), and the recovery of implicit causal references.

Since each CASREP reports on an individual failure event that typically has only a single root cause, and the message writer will only leave a causal reference implicit if its recovery requires simple inferencing by an expert reader, extremely deep domain knowledge (such as would be needed for diagnosis, a related but different task) is not necessary in the information extraction task. The purely structural information that can be gleaned from equipment manuals and parts lists, combined with a set of heuristic inferencing rules, works correctly in the majority of cases. Extending this knowledge in the direction of a behavioral (simulation) model, however, does increase a natural language understanding system's accuracy and robustness as well as its possible applications.

References

[1] D. A. Dahl, "Focusing and Reference Resolution in PUNDIT," *Proceedings of the AAAI*, Philadelphia, PA, Aug. 11-15, 1986, pp. 1083-1088.

[2] J. deKleer and J. S. Brown, "A Qualitative Physics Based on Confluences," *Artificial Intelligence*, 1984, pp. 7-83.

[3] R. Grishman, "PROTEUS Parser Reference Manual," PROTEUS Project Memorandum #4, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, July 1986.

[4] G. Hirst, *Semantic Interpretation and the Resolution of Ambiguity.* Great Britain: Cambridge University Press, 1987.

[5] T. Ksiezyk and R. Grishman, "Equipment Simulation for Language Understanding," *International Journal of Expert Systems Research and Applications*, Vol. 2, No. 1, 1989, pp. 33-78.

[6] E. Marsh, H. Hamburger and R. Grishman, "A Production Rule System for Message Summarization," *Proceedings of the AAAI*, Austin, TX, Aug. 6-10, 1984, pp. 243-246.

[7] E. Marsh, J. Froscher, R. Grishman, H. Hamburger and J. Bachenko, "Automatic Processing of Navy Message Narrative," NRL Report 8893, Aug. 1985.

[8] N. Sager, *Natural Language Information Processing: A Computer Grammar of English and its Applications.* Reading, MA: Addison-Wesley, 1981.

[9] R. Schank and R. Abelson, *Scripts, Plans, Goals and Understanding.* Hillsdale, NJ: Lawrence Erlbaum, 1977.

[10] K. Wauchope, M. K. DiBenigno and E. Marsh, "Automated Text Highlighting of Navy Equipment Failure Messages," NRL Report 9154, Nov. 2, 1988.